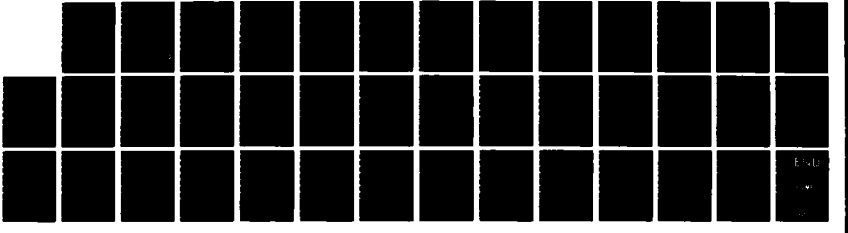
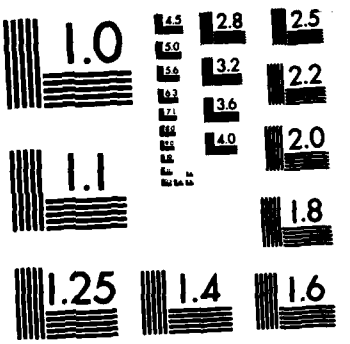
AD-A160 221 METHODS OF CONCEPTUAL CLUSTERING AND THEIR RELATION TO NUMERICAL TAXONOMY (U) CALIFORNIA UNIV IRVINE DEPT OF INFORMATION AND COMPUTER SCIEN... D FISHER ET AL 1/1

UNCLASSIFIED 22 JUL 85 UCI-ICS-85-26 N00014-84-K-0345 F/G 12/1 NL





(12)

Information and Computer Science

Methods of Conceptual Clustering and their Relation to Numerical Taxonomy

Douglas Fisher
Pat Langley

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California
Irvine, California 92717

- TECHNICAL REPORT -

UNIVERSITY OF CALIFORNIA
IRVINE

85 10 11 023

Methods of Conceptual Clustering and their Relation to Numerical Taxonomy

Douglas Fisher
Pat Langley

Irvine Computational Intelligence Project
Department of Information and Computer Science
University of California
Irvine, California 92717

Technical Report 85-26

July 22, 1985

We thank Dennis Kibler and Dan Easterlin of UCI for fruitful discussions on some of the issues addressed in this paper, as well as several participants of the AT&T Artificial Intelligence and Statistics Workshop (1985), including Jim Corter, Tom Ellman, David Hand, Mitch Marcus, Sande Pruzansky, and Bill Gale.

To appear in *Artificial Intelligence and Statistics*, Addison-Wesley, 1985.

This research was supported by Contract N00014-84-K-0345 from the Information Sciences Division, Office of Naval Research. Approved for public release; distribution unlimited. Reproduction in whole or part is permitted for any purpose by the United States Goverment.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Technical Report No. 3	2. GOVT ACCESSION NO.: AD-A160221
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Methods of Conceptual Clustering and Their Relation to Numerical Taxonomy	5. TYPE OF REPORT & PERIOD COVERED: Interim Report 1/85 - 6/85
6. PERFORMING ORG. REPORT NUMBER: 85-26	
7. AUTHOR(s): Douglas Fisher and Pat Langley	8. CONTRACT OR GRANT NUMBER(s): N00014-84-K-0345
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Information and Computer Science, University of California, Irvine, California 92717	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Information Sciences Division, Office of Naval Research, Arlington, Virginia 22217	12. REPORT DATE: July 22, 1985
	13. NUMBER OF PAGES: 34
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

To appear in Artificial Intelligence and Statistics, Addison-Wesley, 1985

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

conceptual clustering, numerical taxonomy,
concept formation, heuristic search,
hierarchical classification, exploratory data analysis.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

ABSTRACT

Artificial Intelligence (AI) methods for machine learning can be viewed as forms of exploratory data analysis, even though they differ markedly from the statistical methods generally connoted by the term. The distinction between methods of machine learning and statistical data analysis is primarily due to differences in the way techniques of each type represent data and *structure* within data. That is, methods of machine learning are strongly biased toward *symbolic* (as opposed to numeric) data representations. We explore this difference within a limited context, devoting the bulk of our paper to the explication of *conceptual clustering*, an extension to the statistically based methods of numerical taxonomy. In conceptual clustering the formation of object clusters is dependent on the quality of 'higher-level' characterizations, termed concepts, of the clusters. The form of concepts used by existing conceptual clustering systems (sets of *necessary and sufficient conditions*) is described in some detail. This is followed by descriptions of several conceptual clustering techniques, along with sample output. We conclude with a discussion of how alternative concept representations might enhance the effectiveness of future conceptual clustering systems.

Table of Contents

1. Introduction

Exploratory data analysis can be characterized as a search for *regularity* or *structure* among objects in an environment, and the subsequent *interpretation* of discovered regularity. At this level of abstraction, many Artificial Intelligence (AI) methods for machine learning qualify as techniques for exploratory data analysis, even though they differ markedly from the statistical methods generally connoted by the term.

In the traditional (statistical) form of exploratory data analysis, numeric summaries of data are the most common means of representing structure in the data. Hartwig and Dearing [HART79] assert that when operating within an *exploratory mode* of data analysis, the analyst must be open to the possibility of several alternative, but equally legitimate, structures in the data. They argue that this openess is best facilitated when the analyst does not place excessive trust in numeric summaries of data, but utilizes visual displays of data as well. AI is also biased against numeric summaries as the only means of data representation, albeit much more so than Hartwig and Dearing. *Symbolic* representations play the predominant role of data representation in AI generally and in machine learning specifically.

Thus one difference between statistical exploratory data analysis and machine learning lies in the representational systems each field uses for representing data and structure within data (numeric vs. symbolic). We shall explore this difference within a limited framework. The bulk of our paper is devoted to the explication of *conceptual clustering*, originally motivated and defined as an extension to methods of *numerical taxonomy* [MICH80]. The purpose of both numerical taxonomy and conceptual clustering methods is to form classification schemes over an initially unclassified set of data. Our explication of conceptual clustering will include descriptions of five conceptual clustering programs, and will mainly serve to illustrate how data and structure within data are represented in machine learning processes, and how search for structure is controlled within the body of a machine learning program.

2. Numerical Taxonomy and Conceptual Clustering

The task of both numerical taxonomy and conceptual clustering methods (ie. any clustering algorithm) is to construct a classification scheme over some set of objects. To this end, a clustering algorithm utilizes a function which measures the *similarity* between objects and/or groups of objects. The abstract clustering task may be defined as follows:

The Abstract Clustering Task

Given: A set of objects, O.

Goal: Distinguish *clusters* (i.e., subsets of O) $s_1, ..., s_n$, such that intra-cluster object similarity of each s_i tends to be maximized, and the inter-cluster object similarity over all s_j's tends to be minimized. A collection of clusters is termed a *clustering.*

Michalski [MICH80] distinguishes methods of conceptual clustering and numerical taxonomy within the above abstraction based on the form of their respective similarity functions. Our development and definition of conceptual clustering to follow draws significantly upon discussion by Michalski [MICH80].

2.1 Numerical Taxonomy

In methods of numerical taxonomy [EVER80], the similarity between two objects is the value of a numeric function applied to the descriptions of the two objects. The description of an object is a vector of variable values, where quantitative, nominal (categorical), and binary-valued variables may be allowed. A data analyst is typically responsible for computing the pair-wise similarity of all objects in a data set and for inputing a matrix of these similarities to a numerical taxonomy program. The similarity matrix is then used by the program to group objects which tend to be most similar, and distinguish objects which are least similar. Intra-cluster and inter-cluster similarity are computed by a function of the pair-wise similarities of the objects in each cluster. Given two objects, A and B, with descriptions, A′ and B′, a typical similarity measure between A and B has the form

$$\text{Similarity}(A, B) = f(A', B')$$

Such a similarity measure is termed *context-free*, since the similarity between A and B is independent of A's and B's relationship to other objects being clustered. *Context-sensitive* measures of similarity have also been developed, in which the similarity of two objects is dependent on their relation to additional objects. That is, within a set of objects, O, with a set of symbolic descriptions, O′, the similarity of two objects, A and B, has the form

$$\text{Similarity}(A, B) = f(A', B', O')$$

If we assume integers are 'objects', then using a context-sensitive similarity measure, the integers 1 and 9 would be considered more similar when considered within the range 1 to 100 than when considered within the range 1 to 10.

Using a numerical taxonomy program, the data analyst may guide the search for useful classification schemes by standardizing the raw data in a number of ways,

and/or by using different similarity functions to build the similarity matrix input to the program.

Within the literature on numerical taxonomy, several classes of techniques have been identified, three of which we shall briefly discuss here:

Optimization techniques attempt to form an optimal K-partition over an object set (i.e., divide the object set into K mutually-exclusive clusters) where K is supplied by the user. Optimization techniques make an extensive search for an optimal K-partition, making them computationally expensive and constraining their use to small data sets and/or small values of K.

Hierarchical techniques form binary classification trees, termed *dendograms*, over object sets. Leaves of the tree represent individual objects, and internal nodes represent object clusters. Hierarchical techniques can be further divided into *agglomerative* and *divisive* techniques, which construct the dendogram bottom-up and top-down, respectively. Hierarchical techniques are computationally cheaper than optimization techniques.

Clumping techniques return clusterings in which constituent clusters may overlap. The possibility of overlap stems from independently considering some number of clusters as possible hosts for an object. A problem with some clumping techniques is that several renditions of the same object set may be obtained.

2.2 Conceptual Clustering

Despite the usefulness of numerical taxonomy techniques, any such method (whether it uses context-free or context-sensitive measures) suffers from a major limitation - the resultant clusters may not be easily characterized in a generalized conceptual language. This limitation can be of concern to a data analyst (or learning program) who (which) wishes to abstract the underlying conceptual structure of object clusters in order to hypothesis about future observations. In conceptual clustering, we do not want to represent a cluster as simply an extensional enumeration of objects, but intensionally, by rules which define membership. We term a collection of these rules, a *concept*. Conceptual clustering is a process abstraction defined by Michalski [MICH80], which addresses the problem of determining conceptual representations of object clusters. Given a set of concepts, C, which may be used to describe structures within an object set, O, Michalski defines the similarity between two objects, A and B, as

$$\text{Similarity}(A, B) = f(A', B', O', C)$$

In other words, the similarity between two objects is dependent on the quality of concepts used to describe the two objects. Extending this idea, the quality of an object cluster is dependent on the quality of concepts which describe the cluster. Definitions of concept quality will vary from program to program and in the second half of the paper we will formalize the notion of quality for individual programs. For now however, to clarify the distinction between methods of numerical taxonomy and conceptual clustering, consider the object set given in figure 1. Each object is defined along two variables, V_1 and V_2, each with values ranging from 0 to 1.

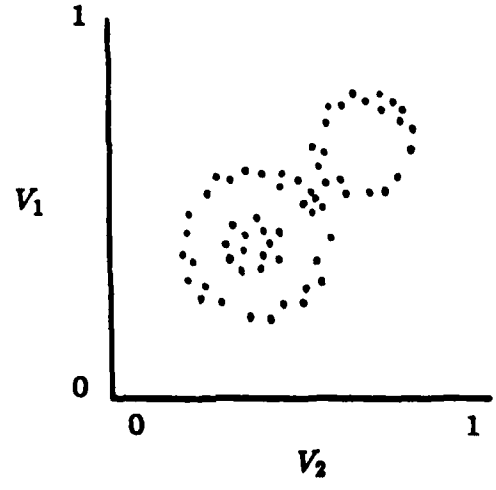


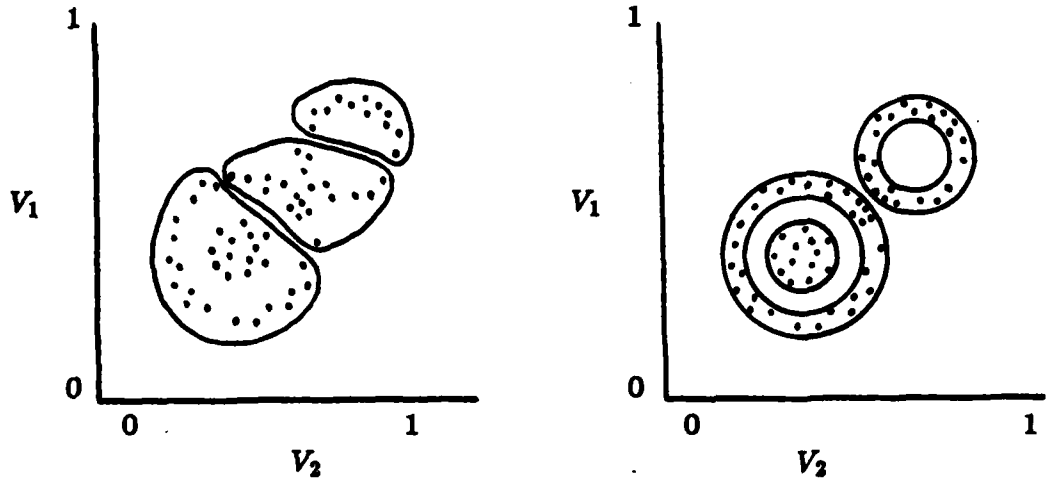
FIGURE 1 - Object Set Displayed in 2 Dimensions

In methods of numerical taxonony, a reasonable similarity measure would employ the inverse of the spatial distance between objects as represented in 2-space. A group of object clusters which maximize some function of intra-cluster similarity and inter-cluster similarity would then be chosen as a clustering. In conceptual clustering, objects are grouped so as to maximize the quality of concepts used to describe clusters. For this example we will assume that concepts have the form

$$r_1 \leq \sqrt{(v_1 - c_1)^2 + (v_2 - c_2)^2} \leq r_2$$

Graphically interpreted, we assume the conceptual clustering algorithm groups objects into clusters which form rings. To do so, the algorithm must identify appropriate constants, r_1, r_2, c_1, and c_2, so as to maximize the quality of derived

concepts.[1] In this example we might assume a concept quality function which measures several factors, one of which is the difference between r_1 and r_2 (i.e., the width of a ring). Possible clusterings obtained by a numerical taxonomy method and a conceptual clustering method are given in figure 2.[2]



Numerical Taxonomy Conceptual Clustering

FIGURE 2 - Possible Clusterings Obtained by a Numerical Taxonomy Method and a Conceptual Clustering Method

It should be clear from the example, that by restricting the possible concepts that a conceptual clustering algorithm can manipulate, we also restrict the set of possible clusterings which can be constructed. Ideally, we would like to endow a conceptual clustering program with a significant body of possible concepts, and allow the program to perform the *search* necessary to extract conceptual structure from a set of objects. We will discuss the search process next.

[1] So as not to mislead the reader, we should note that present conceptual clustering algorithms cannot manipulate conceptual forms as complex as our example, though current research is addressing this limitation. In section 3.0 we will discuss the form of concepts handled by present techniques.
[2] Michalski and Stepp [MIC83A] present further examples contrasting their conceptual clustering method with specific methods of numerical taxonomy.

2.2.1 Two Processes in Conceptual Clustering

In conceptual clustering, we are not only interested in identifying object groups (clusters) as in numerical taxonomy, but in identifying higher level characterizations (conceptual descriptions) of object groups, and using these characterizations to guide the search for a set of 'best' object groups. Thus, two problems must be addressed in conceptual clustering.

- The *aggregation* problem involves determining useful subsets of an object set. Thus, it consists of identifying a set of object classes, each defined as an extensionally enumerated set of objects.
- The *characterization* problem involves determining a useful characterization (concept) for some (extensionally defined) object class, or for each of multiple object classes.

A natural approach to solving the conceptual clustering problem is to first solve the aggregation problem, and then the characterization problem. In machine learning, the characterization problem has been extensively addressed, and is known as the problem of *learning from examples.* Given a number of object sets, the task of learning from examples involves identifying one or more conceptual descriptions for each object set. Methods for learning from examples may be viewed as conducting a search through a *space of concepts* for each object set [MITC82]. For each concept reached in the search, one must evaluate the concept as to whether it usefully describes the object set under consideration.

Most of the current conceptual clustering methods exploit well-understood methods for learning from examples, by making such a process subordinate to a higher-level aggregation process. That is, one searches through a space of clusterings by first *generating* some number of possible clusterings. For each clustering generated, one calls a learning from examples subroutine, which generates a number of possible conceptual descriptions for the clustering. The 'quality' of each of these conceptual descriptions is then *evaluated*, and one (or more) 'best' description(s) is returned by the learning from examples subroutine. The conceptual description of each clustering, passed up from the learning from examples subroutine, is then used in the evaluation of the quality of each clustering, and a 'best' clustering may then be selected. We illustrate this two-tiered search process in figures 3 and 4.

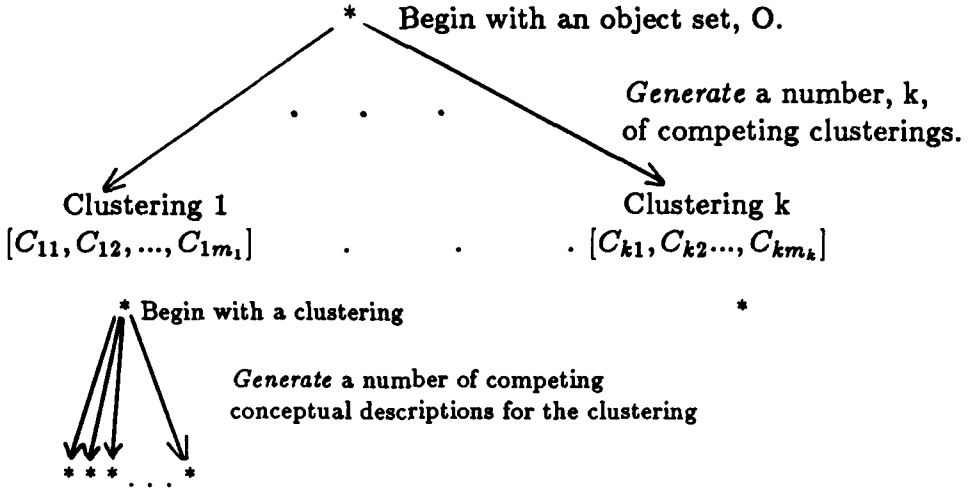


FIGURE 3 - Generation Phase of the Conceptual Clustering Search Process

In describing a number of conceptual clustering techniques, we will focus our discussion on how each technique generates and evaluates object clusterings. This will entail describing the form of concepts which can be used to describe object clusters (section 3.0), but we will not discuss how such concepts are derived. For explication of processes of concept derivation, the interested reader is directed to the literature on machine learning from examples. A very readable account is given by Mitchell [MITC82].

2.2.2 Types of Conceptual Clustering Techniques

One can impose a classification scheme over methods for conceptual clustering similar to that given for numerical taxonomy techniques. Specifically, in surveying conceptual clustering techniques, we will consider *optimization*, *hierarchical*, and *clumping* methods for conceptual clustering.

Optimization techniques of conceptual clustering attempt to construct an optimal K-partition (i.e., K mutually-exclusive clusters) over an object set, where K is supplied by the user. In optimization methods of conceptual clustering, as with methods of numerical taxonomy, the clusters of a constructed partition must be mutually-disjoint with respect to the observed objects. Further, concepts used to describe object clusters must themselves imply object classes which are mutually-disjoint (i.e., disjoint with respect to unobserved or theoretically possible objects). Figures 3 and 4 (above), illustrate how the search for possible partitions and the subordinate search for concepts might interact in an optimization technique of conceptual clustering. We will discuss one optimization technique of conceptual clustering in section 4.0.

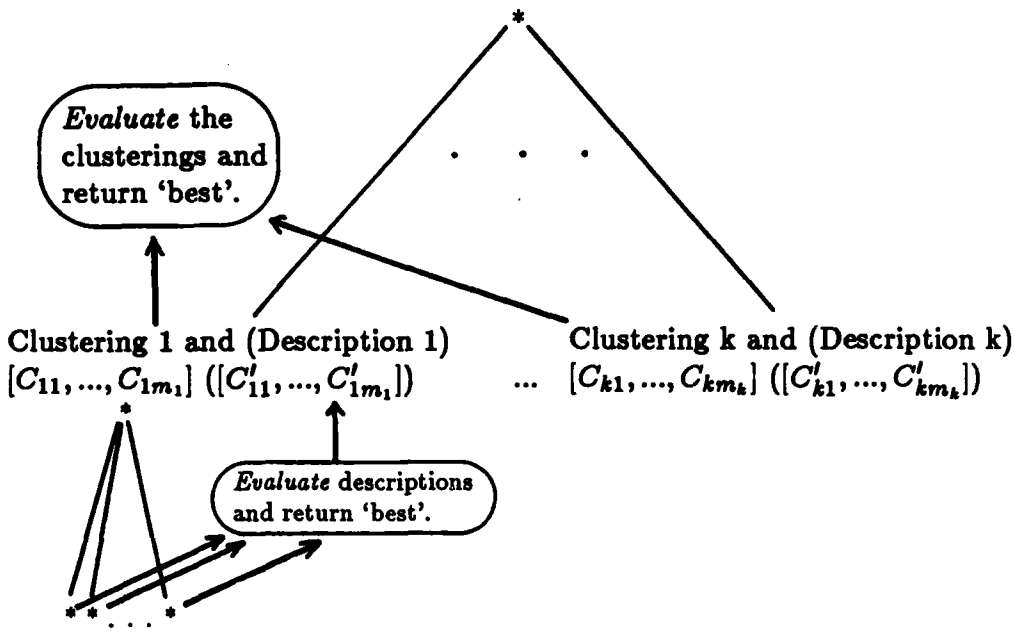


FIGURE 4 - Evaluation Phase of Conceptual Clustering Search Process

Hierarchical techniques of conceptual clustering form classification trees over an object set. Each node in the classification tree, including leaves, represents an object class. Arcs in the tree are labelled by concepts describing these classes. We will present three conceptual clustering hierarchical techniques in section 4.0. Each of these methods constructs a classification tree top-down; in other words, each is a divisive technique (this does not exclude the possibility of agglomerative conceptual clustering methods). In constructing a classification tree, each of our example techniques must partition object classes representing nodes in the classification tree, and ascribe concepts to partition elements. In divisive hierarchical techniques, the division of a node can be framed as a search for partitions combined with a subordinate search for concepts describing clusters of competing partitions, just as with optimization methods. Division of individual nodes occurs within the larger process of classification tree construction, leading us to describe hierarchical techniques as conducting a three-tiered search: a search through a space of hierarchies; a search through a space of partitions; and a search through a space of concepts.

Clumping techniques of conceptual clustering construct classification schemes in which the concepts derived for describing clusters imply possibly overlapping object classes. In section 4.0 we will discuss one conceptual clumping technique which constructs hierarchical, graph-structured classifications.

Before we begin our survey of conceptual clustering methods, we will discuss the general form of objects and concepts used by present conceptual clustering techniques. For readers not familiar with AI representations, this will serve to introduce one restricted form of object and concept representation, and will introduce terminology we use in the remainder of the paper.

3. More on Objects and Concepts

Conceptual clustering programs to date have represented objects as sets of variable-value[3] pairs. All of the conceptual clustering methods we will examine allow objects to be described in terms of nominal or categorical variables, the domains of which are a finite set of discrete values.[4] We present some examples of variables and their domains in table 1; we will be using these variables in examples throughout this section.

Variables	Domains
Color	{blue, red, green}
Size	{large, medium, small}
Shape	{sphere, block, wedge}

TABLE 1 - Some Example Variables and Domains

As we have seen, one of the main components of the conceptual clustering process involves characterizing object clusters. A conceptual clustering program is given a set rules or *operators* which can be used to generate concepts from a set of object descriptions. To ease the process of generating concepts from objects, concept and object representations are typically defined within the same formalism. This implies that all object representations are concept representations, but not vice versa. For the programs we examine, a concept is equivalent to a set of *varible-value set* pairs.[5] An object is a concept in which the value set of each variable is a singleton.

---

[3] *Variable* is synonymous with *attribute*.

[4] In addition, two methods by Michalski and Stepp [MIC83A, MIC83C] allow integer-valued variables and *structured* variables, the domains of which are tree-structured. That is, a classification hierarchy is defined over the values of a structured variable.

[5] Many machine learning programs use more complex concept representation languages. *Relational* or *structured* representations [NILS80] allow one to describe relations between variables. An instance of a relational representation is the concept form given in conjunction with figure 1.

Consider the following concept.

{ [Color = {blue,red}], [Size = {large}], [Shape = {sphere,block}] }

This concept is a 'generalization of' any set of objects which are blue or red in color, *and* are large in size, *and* have a block or sphere shape. We will say that a concept is a *generalization of* an object set if the value set of each variable in the concept includes each object's value for that variable.[6] Similarly, we say an object is a *member* of a concept if the object's values along each variable are in the concept's value set for that variable. Implicit in these definitions is the assumption that all concepts and objects are defined by the same variables. Knowing this, a 'short-hand' representation for a concept is to omit a variable from the concept if the variable's value set in that concept is the domain of the variable. In other words, if a variable is not explicitly given in a concept representation, then this ommision is interpreted as meaning that a member of this concept may possess *any* value of the ommitted variable. Thus, we are *dropping conditions* which are not relevent to defining concept membership. This definition of concept is similar to, but more general than, the definition of a *conjunctive concept* found in [BRUN56].

Given the concept language presented, one can generate concepts which are generalizations of an object set by generating value sets which include the values of all objects along each variable. For the variable *Color*, whose domain is given in table 1, consider the possible generalizations over the value sets of table 2.

Value Sets	More General Value Sets
{blue},{red}	{blue, red} or {blue, red, green}

TABLE 2 - Possible Generalized Value Sets

One may obtain concepts by combining appropriate value sets for distinct variables. Consider the object set in table 3 along with three concepts which are generalizations of this set.

[6] When we state that a concept is a generalization of an object set, we are refering to a property of the concept, and *not* to the process which generated the concept. Concept generation may employ *specialization* operators, as well as *generalization* operators.

Object Set

{ [Color={blue}],[Size={large}],[Shape={sphere}] }
{ [Color={blue}],[Size={medium}],[Shape={sphere}] }
{ [Color={blue}],[Siz={small}],[Shape={block}] }

Three Generalizations of the Object Set

1) { [Color={blue}], [Size={large,medium,small}], [Shape={sphere,block}] }
or

2) { [Color={blue}], [Size={large,medium,small}], [Shape={sphere,block,wedge}] }
or

3) { [Color={blue,red,green}], [Size={large,medium,small}], [Shape={sphere,block,wedge}] }

TABLE 3 - An Object Set and Three Generalizations of the Set

By *dropping conditions* we can reexpress the concepts of table 3 in the following equivalent forms.

1) { [Color={blue}],[Shape={sphere,block}] }

2) { [Color={blue}] }

3) { }

Notice that although each concept is a generalization of the object set, concept 3 is 'more general than' concepts 1 and 2, and similarly, concept 2 is 'more general than' concept 1. It is apparent that concepts can be characterized by their degree of generality with respect to the object sets they describe. That is, concepts are *partially ordered* by the relation *more general than.*

Definition 1
A concept, C_i is more general than a concept, C_j, if all variable value sets of C_j are proper subsets of the corresponding value sets of C_i. If this is the case, then we can also say that C_j is less general than C_i.

At the bottom end of the generality scale are those concepts of least generality or *maximal-specificty.*

Definition 2
A concept, C_i, is a *maximally-specific concept* of an object set, if C_i is a generalization of the object set, and there is no other generalization of the object set which is less general than C_i.

In the concept language we have been discussing, there is exactly one maximally-specific concept for any object set.[7] For example, the only maximally-specific concept of the set of objects given in table 3 is

{ [Color={blue}], [Shape={sphere,block}] }

Means of controlling the generality of concepts describing object clusters are required if useful concepts are to be obtained. For instance, a concept in which all 'conditions' have been 'dropped' does not enlighten us as to the logical correlations which exist among values over an object set. Clumping techniques, which allow concepts which imply overlapping object classes, must especially guard against overly general concepts.[8] On the otherhand, concepts formed by techniques which insist on mutually-disjoint clusters (i.e., optimization and hierarchical techniques), are bounded in terms of their degree of generality. Optimization and hierarchical methods must devise concepts which discriminate the objects of one cluster from objects of every other cluster. These methods must form *discriminant concepts.* A concept is a *discriminant* concept of an object set, O, with respect to another object set, Q, if all objects of O are members of the concept, and no member of Q is a member of the concept.

Concepts formed by hierarchical and optimization techniques are bounded above in their generality by *maximally-general discriminant concepts.*

Definition 3
A concept, C, is a *maximally-general discriminant concept* of an object set, O, with respect to a set, Q, iff C is a discriminant concept of O with respect to Q, and there is no other discriminant concept of O with respect to Q, which is more general than C.

Consider the following example of two object classes and associated maximally-general discriminant concepts.

Class 1
{ { [Color={blue}], [Size={large}], [Shape={sphere}] } }

Class 2
{ { [Color={red}], [Size={large}], [Shape={block}] },
{ [Color={red}], [Size={large}], [Shape={wedge}] } }

[7] Concept languages less restrictive than the one we have assumed (eg. relational concepts) will allow multiple maximally-specific concepts per object set.

[8] Recall that a problem with clumping techniques of numerical taxonomy was that object classes could be multiply defined. An analogous problem with conceptual clumping methods might be the construction of concepts which imply the same object classes.

Two maximally-general discriminant concepts of Class 1 with respect to Class 2 are given below.

1) { [Color={blue,green}] }

2) { [Shape={sphere}] }

Additionally, we can give two maximally-general discriminant concepts of Class 2 with respect to Class 1.

1) { [Color={red,green}] }

2) { [Shape={block,wedge}] }

Given maximally-general discriminant descriptions of Class 1 with respect to Class 2, and vice versa, there are 4 ways to assign maximally-general discriminant concepts to classes 1 and 2.

		Class 1 Concepts	Class 2 Concepts
Combination implying mutually-disjoint classes	1)	{[Shape={sphere}]	[Shape={block,wedge}] }
	2)	{[Color={blue,green}]	[Color={red,green}] }
Combinations implying overlapping classes	3)	{[Color={blue,green}]	[Shape={block,wedge}] }
	4)	{[Shape={sphere}]	[Color={red,green}] }

TABLE 4 - Combinations of Maximally-General Discriminant Concepts

Notice that although each of the above combinations perfectly distinguish the objects of class 1 from class 2, and vice versa, only the first combination implies a partition over the set of theoretically possible objects. The first combination implies 2 mutually-disjoint clusters, because membership in both clusters is based on non-overlapping values along the same variable. In general, non-overlapping value sets of the same variable will imply non-overlapping clusters, and each value set (when interpreted as a concept with all other value sets dropped out) will constitute a maximally-general discriminant concept of the object group it implies, with respect to all object groups implied by other value sets. This observation is central to the processing of two hierarchical systems, DISCON and RUMMAGE, discussed in the next section. The latter three combinations of table 4 imply overlap with respect to unobserved objects (e.g., consider any green object, a blue block, and a red sphere).

Typically, it will be the case that for some number of object classes, there will be no assignment of maximally-general discriminant concepts to each object class, with respect to the remaining object classes, in a way that completely avoids overlap. This point has ramifications for the processing of the CLUSTER/2 system, which is discussed in the next section.

We have now developed the necessary ideas and terminology for discussing a number of conceptual clustering systems.

4. Some Conceptual Clustering Algorithms

In this section we survey a number of conceptual clustering algorithms. This survey includes one optimization technique, three hierarchical techniques, and a clumping technique. In discussing these techniques we stress how each technique solves the *aggregation problem* and how each method evaluates clustering quality. Given our discussion in section 3.0, we will abstract out most of the detail concerning how each method solves the *characterization problem*, that is, the process they use to obtain concepts for describing object clusters.

4.1 The Partitioning Module of CLUSTER/2

The Partitioning Module of CLUSTER/2 by Michalski and Stepp [MIC83A, MIC83C] is an optimization technique of conceptual clustering. Given an object set and a user-supplied value, K, the Partitioning Module attempts to construct an optimal K-partition over the object set. CLUSTER/2 allows objects and concepts to be defined in terms of nominal, integer, and structured variables. For the sake of clarity, we will assume only nominal variables, in considering examples. Given an object set, O, and a partition size, K, the CLUSTER/2 algorithm may be outlined as follows.

1) **Construct a number of initial clusterings**, each with K clusters. Each of these alternative clusterings may possess overlapping clusters.

2) **Make each initial clustering disjoint and identify concepts** for each clustering.

3) **Evaluate the quality of each clustering** and select a 'best' initial clustering.

4) **Continue the search for an optimal clustering** by 'modifying' the best initial clustering.

4.1.1 Constructing Initial Clusterings

Given the task of forming a K-partition over an object set, the Partitioning module intially selects K *seed* objects at random. Intuitively, each seed will act as a cluster center. The system treats each seed as a member of a singleton object class (i.e., there are K object classes with one member each). The program then derives maximally-general discriminant concepts for each seed class with respect to all other seed classes. The result is that for each seed, *a number of* maximally-general concepts are derived which cover that seed and no other seed. Each concept of each seed class also covers some number of non-seed objects. That is, each concept implies an object class which contains one seed and multiple non-seed objects. We illustrate the above process in figure 5.

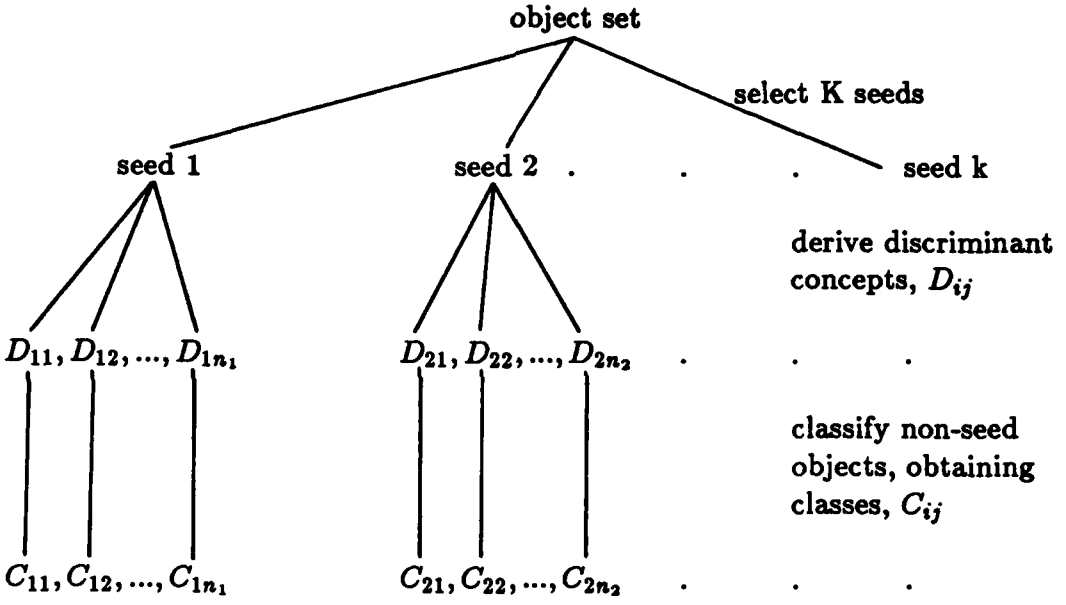


FIGURE 5 - Creating Initial Clusterings in CLUSTER/2

By combining object classes, C_{li}, implied by maximally-general concepts describing distinct seeds, D_{li}, we obtain a clustering which is guarenteed to classify all objects (seed and non-seed). At termination, the process described above has constructed a number of possible clusterings, each having the form $C_{1i_1}, C_{2i_2}, ..., C_{ki_k}$. However, each of the possible clusterings may possess overlapping clusters, with respect to non-seed objects. The following process seeks to make these clusters mutually-disjoint and assign conceptual descriptions to the non-overlapping clusters.

4.1.2 Describing Object Classes

The construction of maximally-general discriminant concepts for each seed object serves to identify object classes (over both seed and non-seed objects), from which a number of competing clusterings are derived. Each of the clusterings may possess overlapping clusters. Each of the competing clusterings is made disjoint by removing objects which are in more than one cluster. These removed objects are placed in an *exceptions list*, and maximally-specific characteristic concepts are derived for the now disjoint clustering. The derivation of maximally-specific concepts serves to reduce the possibility of overlapping clusters with respect to future, as yet unobserved, objects. Objects on the exceptions list are added back into the clustering one at a time. This is done by creating K different versions of the clustering, and incorporating the exceptional object into a different cluster (of which there are K in number) of each version. The K versions are then evaluated (according to criteria discussed shortly) and a 'best' clustering which incorporates the exceptional object is selected. The above process is performed for each of the competing clusterings. At termination a number of competing partitions (with associated exceptions which could not be added without resulting in overlap) have been generated. These competing partitions can now be evaluated and a 'best' partition selected.

4.1.3 Evaluating Quality of Clusterings

CLUSTER/2 uses a number of criteria for measuring clustering quality. Each of these criteria is a function of the maximally-specific concepts which describe the clusters of a clustering. We will briefly discuss three of these criteria.

The *fit* of a set of concepts with respect to the set of clusters they describe is one criterion for evaluating clustering quality. Fit is the ratio of the number of observed objects from which the concepts were derived (i.e., the number of actual objects in the applicable clustering) and the number of theoretically possible objects (observed plus unobserved) which are covered by the concepts. Fit is an example of a criterion which is a function of the map between a set of concepts and the clusters they describe, and is *analogous* to measures of intra-cluster similarity used in numerical taxonomy.

The *simplicity* of a set of concepts is the total number of variables used in each concept (after *dropping conditions*). Simplicity is an example of a criterion which is only a function of concepts, and not the clusters they describe.

The *disjointness* between two concepts is a function of the number of variables in the two concepts whose values do not intersect. The *inter-cluster difference* of a set of concepts is the sum of the disjointness of all pairs of concepts. This is

a criterion which is analogous to a measure of inter-cluster dissimilarity used in numerical taxonomy.

The user is responsible for ordering the criteria in terms of their importance, and for specifying a minimal value that a clustering must possess for each criterion.

4.1.4 Searching for Optimal Partitions

After selecting a 'best' partition by the steps above, the search for an optimal partition continues. This is done by selecting one seed object from each cluster of the selected partition and iteratively applying the above steps to these new seeds. If partition quality is improving from step to step, seeds which represent the central tendency of each cluster are selected. If partition quality does not improve from one step to the next, seeds are drawn from the 'edge' of each cluster. This process of selecting seeds and devising a partition continues for a user-specified number of iterations. The 'best' partition (which may be sub-optimal) found over all iterations is returned by the Partitioning Module. We will illustrate CLUSTER/2's behavior with a simple example.[9]

Table 5 gives a number of variables and their respective domains that are used to describe animals.

Variables	Variable Domains
Body Covering	hair, feathers, cornified skin(corn.skin), moist skin
Heart Chambers	4, imperfect 4(imp.4), 3
Body Temp.	regulated, unregulated
Fertilization	internal, external

TABLE 5 - Variables Describing Animals

A set of animals (objects) is given in table 6.

Assume the task is to construct an optimal 2-Partition over the 5 objects of table 6, in one iteration. Inter-cluster difference is to be the most important criterion in evaluating clustering quality.

[9] This example is 'hand' executed, and is based on our reconstruction of the algorithm from published reports.

		Body Covering	Heart Chambers	Body Tempeture	Fertilization
	mammal	hair	4	regulated	internal
	bird	feathers	4	regulated	internal
objects	reptile	corn. skin	imp. 4	unregulated	internal
	amphibian-1	moist skin	3	unregulated	internal
	amphibian-2	moist skin	3	unregulated	external

TABLE 6 - An Object Set

The first step is to pick two seeds, which we assume will be mammal and reptile from table 6.

	Body Covering	Heart Chambers	Body Temp.	Fertilization
seed 1)	{hair}	{4}	{reg}	{internal}
seed 2)	{corn. skin}	{imp. 4}	{unregulated}	{internal}

TABLE 7 - Two Seed Objects Initially Selected by CLUSTER/2

For each seed, CLUSTER/2 derives maximally-general discriminant concepts, with respect to the other seed, as shown in table 8.

seed 1 concepts	seed 2 concepts
{[Body Cover={hair,feathers,moist skin}]}	{[Body Cover={corn. skin, feathers,moist skin}]}
{[Heart Chambers={4, 3}]}	{[Heart Chambers={imp.4, 3}]}
{[Body Temp.={regulated}]}	{[Body Temp.={unregulated}]}

TABLE 8 - Maximally-General Discriminant Concepts Discriminating 'mammal' and 'reptile'

As the table shows, there are 9 ways to combine these maximally-general discriminant concepts, so as to imply 9 clusterings, whose 2 clusters may overlap. CLUSTER/2 attempts to make each of these clusterings disjoint. We will consider the clustering (over seed and non-seed objects) implied by the following pair of concepts.

seed 1 concept	seed 2 concept
{[Heart Chambers={4,3}]}	{[Heart Chambers={imp.4, 3}]}
{mammal,bird,amphibian-1,amphibian2}	{reptile,amphibian-1,amphibian-2}

Amphibian-1 and amphibian-2 occur in both clusters. These two objects are removed and placed on an exceptions list. Maximally-specific concepts are derived for each of the now disjoint clusters.

	Cluster 1 {mammal,bird}	Cluster 2 {reptile}
maximally-specific concepts	{[Body Cover={hair,feathers}], [Heart Chambers={4}], [Body Temp.={regulated}], [Fertilization={internal}]}	{[Body Cover={corn.. skin}], [Heart Chambers={imp.4}], [Body Temp.={unregulated}], [Fertilization={internal}]}

TABLE 9 - Maximally-Specific Concepts of Two Disjoint Clusters

Amphibian-1 is now added back into the clustering. This is done by adding amphibian-1 to cluster 1 and generating a maximally-specific concept describing the new cluster. This new cluster, along with the unchanged cluster 2, constitutes one possible clustering which incorporates amphibian-1. A second possible clustering is obtained by adding amphibian-1 to cluster 2, deriving a maximally-specific concept for the new cluster, and considering it along with the original cluster 1. The two possible clusterings, each of which incorporates amphibian-1, are given in table 10.

	Clustering A (adding amphibian-1 to first cluster)	Clustering B (adding amphibian-1 to second cluster)
cluster 1	{[Body Cover={hair,feathers,moist skin}], [Heart Chambers={4,3}], [Body Temp.={regulated,unregulated}], [Fertilization={internal}] }	{[Body Cover={hair,feathers}], [Heart Chambers={4}], [Body Temp.={regulated}], [Fertilization={internal}] }
cluster 2	{[Body Cover={corn. skin}], [Heart Chambers={imp.4}], [Body Temp.={unregulated}], [Fertilization={internal}] }	{[Body Cover={corn.skin,moist skin}], [Heart Chambers={imp.4,3}], [Body Temp.={unregulated}], [Fertilization={internal}] }

TABLE 10 - Two Possible Clusterings Which Incorporate Amphibian-1

These two competing clusterings are now evaluated in terms of some criteria. If simplicity were of highest importance, then Clustering A of table 10 would be selected to incorporate amphibian-1, since we can drop the variable, 'Body Tempeture' from the concept representing cluster 1 of this clustering. Since we

have stated inter-cluster difference is of highest importance, however, Clustering B would be selected to incorporate amphibian-1.

Next, amphibian-2 is removed from the exceptions list and incorporated into Clustering B, above. A process similar to the one described for amphibian-1 incorporation would be followed, and the clustering selected for incorporating amphibian-2 follows in figure 6.

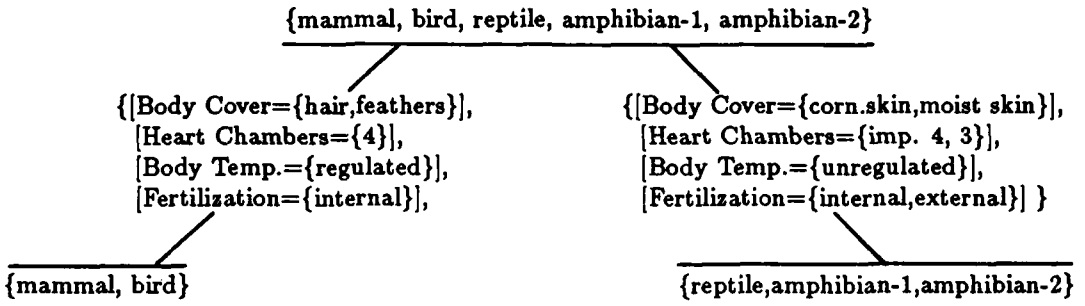


FIGURE 6 - A Partition Formed by CLUSTER/2 Over Objects Given in Table 6

Recall that the process we have just described must be performed for all 9 combinations of maximally-general discriminant concepts given in table 8. In theory each combination could yield a different clustering, and these unique clusterings would then have to be evaluated as to which constituted the 'best' initial 2-partition of the object set. New seed objects would be selected from each cluster of the best partition, thus continuing the search for an optimal partition. However, in our example each of the 9 combinations results in the same clustering, which is illustrated in figure 6, and since we stated we would 'hand' execute only one iteration, this is the final clustering.

As is perhaps evident, the Partitioning Module extensively searches the space of possible partitions, and is computationally quite expensive as a result. Michalski and Stepp report a number of heuristics used to prune the search, but even with these, the Partitioning Module appears to run in time proportional to M^K, where K is the desired partition size, and M is a linear function of the number of defining variables and the average size of all variable domains. CLUSTER/2's extensive search also seems to enable it to effectively handle exceptional objects, and to discover relatively good clusterings, even in ill-structured data.

4.2 The Hierarchy-Building Module of CLUSTER/2

The Hierarchy-Building Module of CLUSTER/2 [MIC83A, MIC83C] contructs a classification tree over an object set. Arc labels in the resultant tree define object classes in terms of multiple variables (ie. the classification is *polythetic*). Tree construction proceeds top-down. The Hierarchy-Building Module employs the Partitioning Module as a subroutine for dividing nodes representing object classes during tree construction. The Hierarchy-Building Module divides an object class, O, (represented by a node in the classification tree), by iteratively calling the Partitioning Module for several small partition sizes (eg. 2,3, and 4), thus producing several partitions of varying sizes. The constituent clusters of the 'best' of these partitions are selected as children of O. The criteria used to compare partitions of equal size in the Partitioning Module are modified to compare partitions of varying sizes in the Hierarchy-building Module. The Hierarchy-building Module constructs a classification tree one level at a time, and tree construction terminates when the clustering represented by the current tree level does not represent an improvement over the clustering representing the previous tree level.

An example classification tree constructed by the Hierarchy-building Module of CLUSTER/2[10] the CLUSTER/2 algorithm based is given in figure 7. The tree was constructed over the data in table 11[11], which represents the 9 animal phyla in terms of 11 variables. The tree was constructed assuming a maximum branching factor of 2.

The Hierarchy-buiding Module calls upon the Partitioning Module to make an extensive search through the space of possible partitions for each node in the classification tree. As a consequence, it inherits the computational expense of the Partitioning Module and thus, for a given maximum branching factor, F, the Hierarchy-building module appears to run in polynomial time of degree F. The Hierarchy-Building Module constructs a single classification tree, and thus does not extensively search the space of trees, but rather depends upon a single good tree emerging from judicious division of individual nodes.

4.3 RUMMAGE

RUMMAGE [FISH84] represents a hierarchical conceptual clustering technique. Like the Hierarchy-building Module of CLUSTER/2, RUMMAGE constructs a classification tree top-down. Unlike trees constructed by CLUSTER/2, the arc-labelling rules of trees constructed by RUMMAGE define object classes in terms of a single variable (i.e., RUMMAGE forms *monothetic* classifications). RUMMAGE allows objects to be defined only in terms of nominal variables.

[10] This tree is the result of 'hand' executing the CLUSTER/2 algorithm as reconstructed from published reports.

[11] This table is taken from [ORAM73]

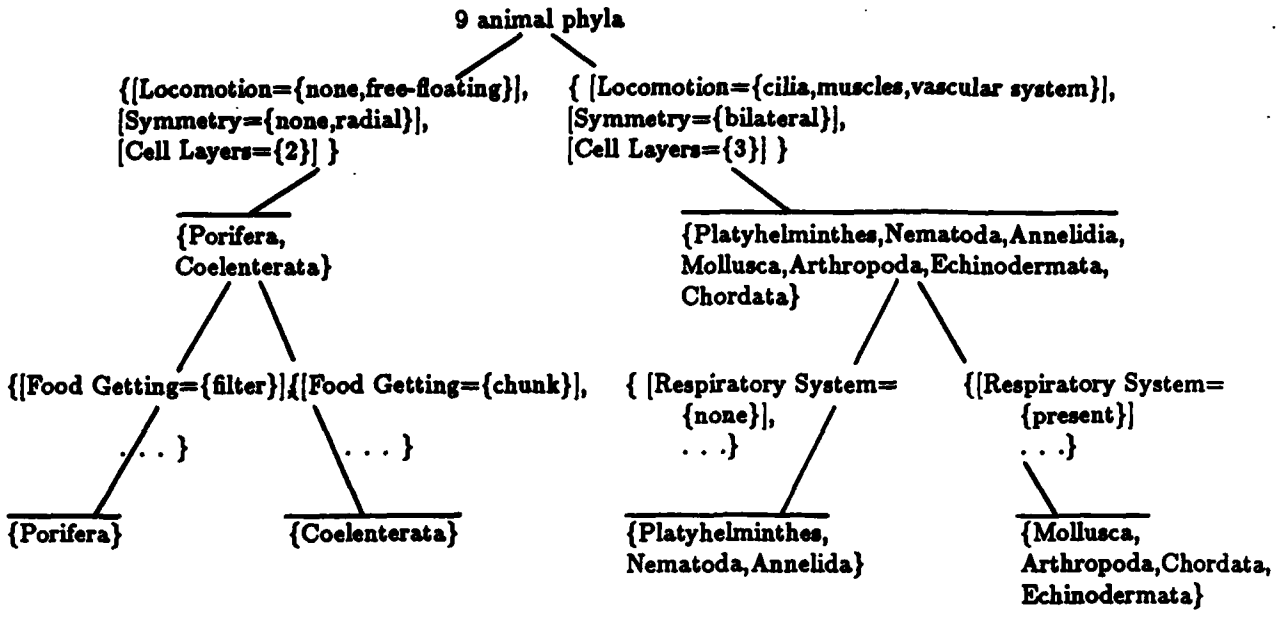


FIGURE 7 - Classification Tree Constructed by CLUSTER/2

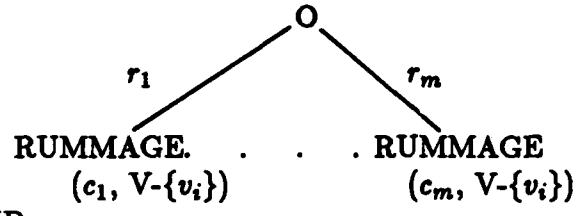
	PORIFERA	COELENTERATA	PLATYHELMINTHES	NEMATODA	ANNELIDA	MOLLUSCA	ARTHROPODA	ECHINODERMATA	CHORDATA
FOOD GETTING	filter feeders	chunk feeders	chunk feeders, parasites	mostly parasites	chunk feeders	chunk feeders	chunk feeders	chunk feeders	chunk feeders
LOCOMOTION	none	mostly sessile; free-floating	muscles, cilia	muscles	muscles	muscles	muscles, appendages	water vascular system	muscles, limbs
SYMMETRY	none or radial	radial	bilateral	bilateral	bilateral	bilateral	bilateral	radial	bilateral
NUMBER OF BODY OPENINGS	one	one	one	two	two	two	two	two	two
NUMBER OF CELL LAYERS	two	two	three	three	three	three	three	three	three
NERVOUS SYSTEM	none	present	present	present	present	present	present	present	present
DIGESTIVE SYSTEM	none	present	present	present	present	present	present	present	present
EXCRETORY SYSTEM	none	none	present	present	present	present	present	present	present
CIRCULATORY SYSTEM	none	none	none	none	present	present	present	present	present
RESPIRATORY SYSTEM	none	none	none	none	none	present	present	present	present
SKELETAL SYSTEM	spicules, no true system	none	none	none	none	hard outer shell	external	mineral deposits	internal

TABLE 11 - Animal Phyla Represented Along 11 Variables

Input to RUMMAGE includes a set of objects, O, and a set of variables, V, over which objects are defined. RUMMAGE selects that variable, v_i, whose values 'best' partition O into mutually-exclusive subsets, o_1 through o_m. RUMMAGE is then recursively called for each object class, o_j. Recursion terminates when RUMMAGE decides there is no variable which produces a partition of user-specified minimal quality. A high-level description of RUMMAGE is given in table 12.

FUNCTION RUMMAGE(O, V)
BEGIN
select a variable, v_i, for which there exists a
partitioning of the domain of v_i, r_1 through r_m,
which implies a 'best' partitioning of O, c_1
through c_m.

IF no v_i is selected THEN RETURN O
ELSE RETURN



END

TABLE 12 - A High Level Description of RUMMAGE

RUMMAGE solves the aggregation problem by forming a number of possible partitions, each of which is implied by the values of a distinct variable. For each partition, p_i, implied by the values of a variable, v_i, RUMMAGE derives a maximally-specific characteristic concept for each cluster of p_i over the remaining variables (ie. all defining variables excluding v_i). Each competing partition is then evaluated in terms of the concepts describing partition clusters. The criteria used for evaluating partition quality are a criterion of *simplicity* and a criterion of *inter-cluster difference*. These criteria are similar in intent to two criteria used by CLUSTER/2, but differ in their details.

An example tree constructed by RUMMAGE over the data of table 11 is given in figure 8.

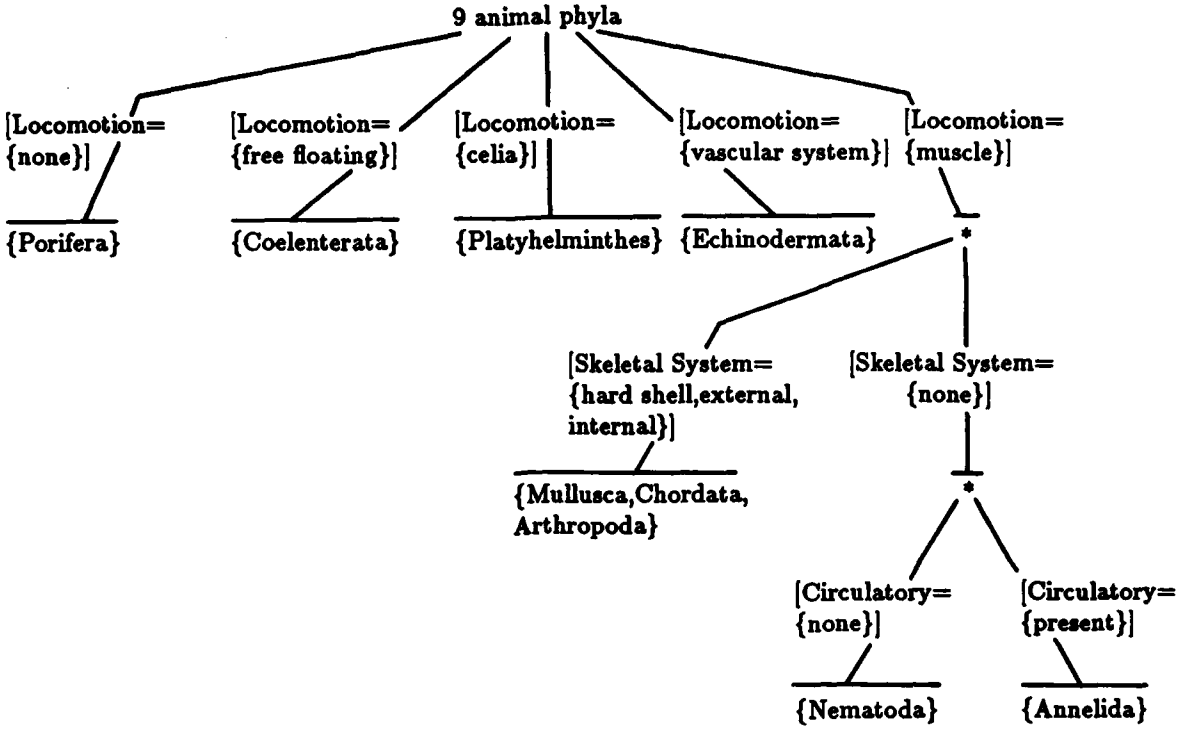


FIGURE 8 - Classification Tree Constructed by RUMMAGE

RUMMAGE is computationally cheaper than the Hierarchy-buiding Module of CLUSTER/2. Like the Hierarchy-building Module of CLUSTER/2, RUMMAGE builds a single tree, and thus depends on a good classification tree emerging from judicial division of individual nodes of the classification tree. However, in dividing an individual node, RUMMAGE searches a much smaller space of possible partitions than does CLUSTER/2. This yields cheaper computation, but also reduces its ability to discover good clusterings in ill-structured data. RUMMAGE is capable of discovering 'good' clusterings implied by single variable values, but in general, it is incapable of discovering good clusterings implied only by a conjunction of values. The 'flattened' top level of the example tree constructed by RUMMAGE is indicative of this limitation. However, one could imagine an extended version of RUMMAGE, in which rather than expanding a node one level at a time, nodes are expanded (some constant) F levels at a time. This extension would require a larger search of possible node divisions (i.e. a search of all subtrees of F levels) which would require polynomial time of degree F (as opposed to linear time for the present implementation). However, the proposed extension would most probably alleviate many of the problems with RUMMAGE discussed above.

4.4 DISCON

Like RUMMAGE, Langley and Sage's DISCON system [LAN84C], employs a top-down, monothetic technique which only allows objects defined over nominal variables. - DISCON solves the aggregation problem in the same way as does RUMMAGE by considering all partitions implied by values of each variable. Unlike RUMMAGE (and CLUSTER/2), DISCON does not derive concepts for object classes and use evaluations of these concepts to guide the search through the space of classification trees. Instead, DISCON constructs all possible monothetic classification trees and selects a tree with a minimal number of nodes. An example of a partial classification tree constructed by DISCON over the data of table 11 is given in figure 9.

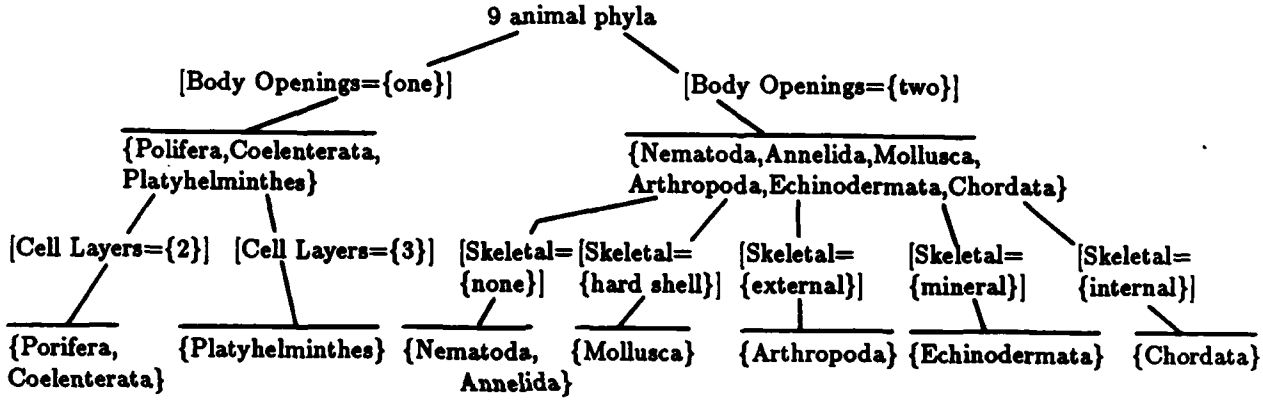


FIGURE 9 - Classification Tree Constructed by DISCON

DISCON carries out a near exhaustive search of the space of possible classification trees for an 'optimal' tree. This makes DISCON computationally quite expensive. The addition of heuristics to DISCON would serve to significantly reduce DISCON's computational requirements. One might, in fact, view RUMMAGE as a heuristic version of DISCON, in which only one tree is built. Recalling our proposed extension of RUMMAGE, in dividing each node of a classification tree, DISCON searches through all divisions (subtrees) of maximal depth.

4.5 UNIMEM

We consider the UNIMEM system by Lebowitz [LEBO82] to be an example of a conceptual clumping program, which (unlike the programs discussed above) constructs clusterings composed of clusters which may overlap. UNIMEM also differs from the above programs in two other important respects. First, UNIMEM expects objects to be presented *incrementally*, as opposed to the other programs discussed, which require an entire object set to be present at the outset of execution. Second, UNIMEM was not explicitly framed by Lebowitz as a conceptual clustering algorithm. Rather, Lebowitz intended UNIMEM to represent a program for *concept formation*,[12] with an intent that the program should represent a reasonable model of human concept formation. UNIMEM was abstracted from an earlier program by Lebowitz, IPP [LEB83A], which has the of task of reading and 'understanding' news stories on international terrorism by building and using a memory of reported terrorist events. Despite Lebowitz' intent, we will present UNIMEM as a conceptual clumping program, and without loss of significant information, impose our terminology for objects and concepts in characterizing its processing.

Objects and concepts are represented identically in UNIMEM, that is, each is represented as sets of variable-value pairs, or in keeping strictly with previously used terminology, sets of variable-value set pairs, where all value sets are singletons. Using this formalism for representing concepts, there is only one means for generating concepts: replace a singleton value set of a variable by the domain of the variable, and employ the previously discussed *dropping conditions* rule. Using concepts of this form, UNIMEM builds a hierarchical clustering in which objects may multiply occur as leaves of the hierarchy. The hierarchical clusterings built by UNIMEM differ from those constructed by CLUSTER/2, RUMMAGE, and DISCON, not only because they allow an object to occur in multiple clusters, but also because UNIMEM distinguishes variable values which label arcs of the hierarchy (as with hierarchical classifications produced by other systems) from variable values which label nodes of the hierarchy. We discuss this distinction next.

UNIMEM labels arcs of the hierarchy with single variable values, which are termed *predictive*. In a clustering, a variable value is predictive of a particular object cluster if the value is shared by all cluster objects and is shared by the objects of very few other clusters. As such, the presence of a predictive value in an object can be used to predict which clusters might incorporate the object. Predictive values are used to constrain the search for clusters which might incorporate an object. UNIMEM considers values as ceasing to be predictive of any cluster if they index more than some user-specified number of clusters. Thus, we may view UNIMEM as using a measure of 'inter-cluster' difference which is a function only of the predictive values over a group of clusters.

[12] In the machine learning literature, conceptual clustering is viewed as a form of concept formation.

In contrast to arc-labelling values, nodes (representing clusters) are labelled by *all* values (predictive and otherwise) common to cluster members. Node labelling values are termed *predictable* values, since their presence in a concept can be predicted from the presence of predictive values of the concept. Note that all predictive values are predictable, but not vice versa. UNIMEM also uses a measure of 'simplicity' which is a function of the predictable values of a cluster. If there are too few predictable values of a cluster, then the cluster is deemed not to represent an important class of objects and it is removed from the clustering.

Last, UNIMEM is given a facility for dealing with exceptional objects. Every predictable value of a cluster has an associated integer weight. The weight of a predictable value of a cluster is decremented when it is not present in an object possessing a predictive value of the cluster. The weight of a predictable value of a cluster is incremented when it is present in an object possessing a predictive value of the cluster. A value is 'dropped' from the concept definition of a cluster if its weight falls below a user-defined threshold, indicating that the value cannot be reasonably predicted from the presence of predictive values.[13]

We will now summarize the discussion above, by outlining the process by which an object is incorporated into a hierarchical clustering. A high-level description of the UNIMEM algorithm is given in table 12. Before any objects are clustered, the clustering is initialized to a single cluster which contains no predictable values.

We now consider an example of UNIMEM's behavior on a simple example. We 'hand' execute UNIMEM (based on our reconstruction) on the object set given in table 6. Assume the constant specifying the maximum number of clusters a predictive value may index is 2. The constant specifying the minimum acceptable weight in order for a variable to be considered predictable is 0 (that is, a value is dropped when its weight reaches -1). The constant specifying the minimum acceptable number of values in a concept is 2. To acquire a clustering of much interest will require that UNIMEM make several iterations through the small number of example objects. For this example, we will iterate through the data set 2 times, a different ordering of the data for each iteration. For the first iteration we assume that the order of object presentation is (amphibian-2, amphibian-1, reptile, bird, mammal). For this example, assume that weights of predictable values are given as superscripts.[14]

[13] In certain circumstances UNIMEM may circumvent this conservative generalization policy and generalise over a concept description and an object if the two share sufficiently many (user-specified) number of values in common. However, we will not detail this process in the discussion to follow.

[14] To simplify this example, we will not consider recursive calls to UNIMEM as specified in step 3b of the UNIMEM algorithm.

Step 1) An object is presented to be incorporated into a hierarchical clustering. UNIMEM first considers what children of the root (i.e., the top node) of the clustering might serve to incorporate the object.

Step 2) A collection is made of all children of the root node which are indexed (predicted) by at least 1 value of the object. Recall that arcs are only labelled by predictive values of clusters.

Step 3) The object is incorporated into the clustering based one of the following rules:

a) If no children of the root were collected in step 2, then make the object a child of the root. This involves directing arcs from the root to the object, each arc labelled by a variable value of the object. All variable values of the object are considered predictive of the object in this situation. This may cause some values to be predictive of more than an acceptable number of clusters, thus causing these values to be removed as predictive of any cluster.

b) If some number of children of the root were collected in step 2, then for each child perform the following:

- Increment all of the child's predictable values which are present in the object.

- If the object has a different value than the child along any variable,

  THEN do each of the following:

  – Decrement the weight of each predictable value which is not present in the object.

  – If the weight of a decremented value falls below a user-specified threshold, then drop this value from the set of predictable values *and* remove this value as a predictive value of the cluster, if the value is in fact, predictive. Removing predictive values of a cluster (and thus the arcs labelled by these predictive values), may cause a cluster to be removed from the clustering if all such predictive values are removed.

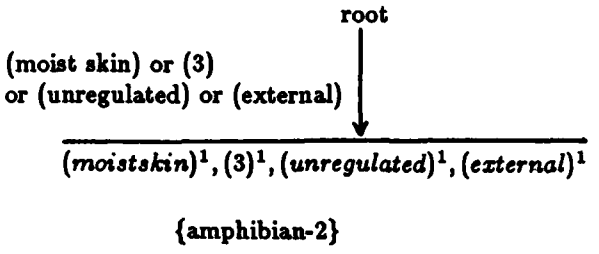
  – If dropping values results in a concept of too few values (according to some user-specified threshold) then remove the concept from the hierarchy, by removing arcs to it from its parent.

  ELSE attempt to incorporate the object into one of the children of the cluster by treating each child as the root of a subordinate clustering and recursively applying steps 1 through 3.

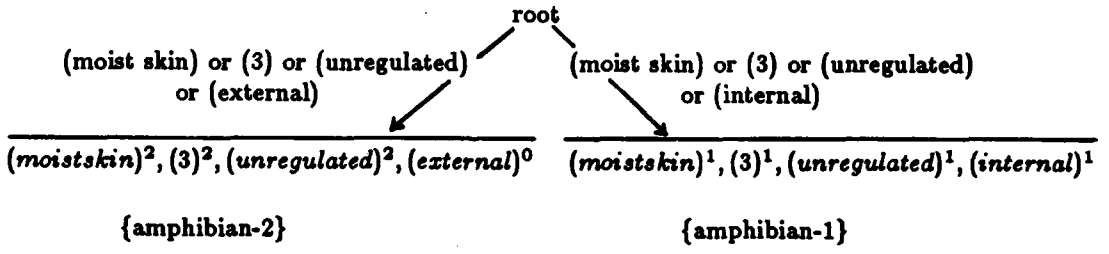
c) If the object could not be incorporated into any cluster in steps a or b above, then make the object a child of the root by the same process as given in step a.

TABLE 13 - The UNIMEM Algorithm

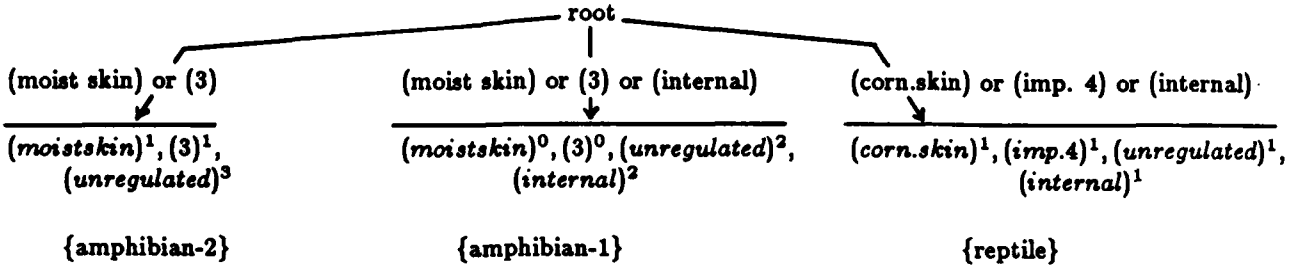
We begin by incorporating amphibian-2 into the initialized clustering. Since there are no children of the root node which might classify amphibian-2, it is simply made a child of the root resulting in the following structure.



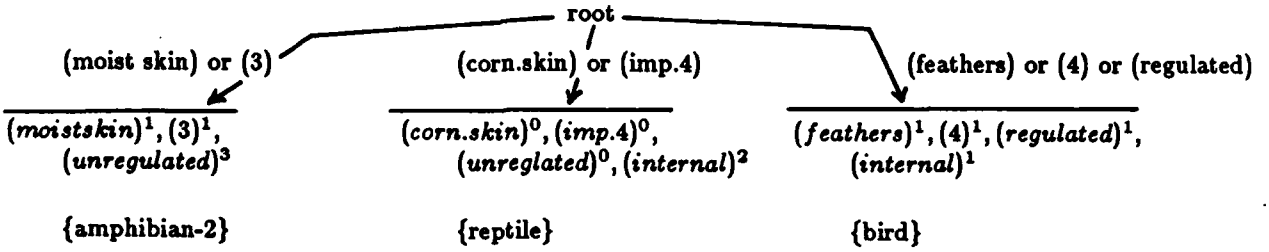
Next amphibian-1 is added to the clustering. Since amphibian-1 has three values which are considered predictive of amphibian-2, amphibian-1 is compared with the values of the concept describing amphibian-2. It is found that amphibian-1 and the concept representing amphibian-2 differ in value along the variable, 'Fertilization', and the weight of the concept's 'Fertilization' value is decremented, while the remaining values are incremented. A concept representing amphibian-1 is then added to the clustering, resulting in the following structure.



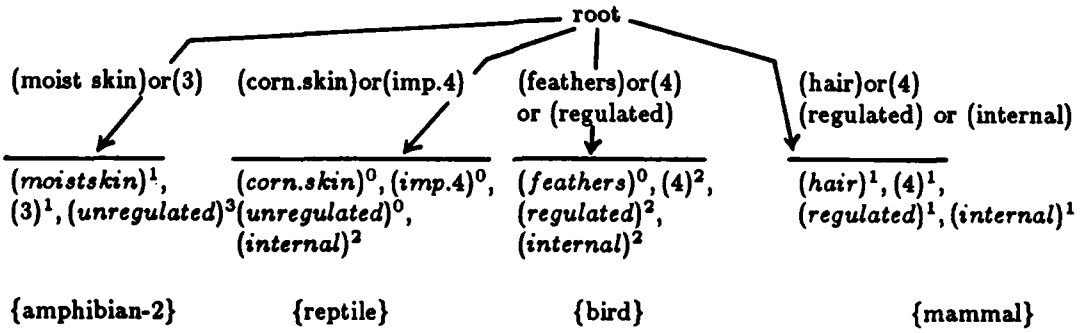
Reptile is now incorporated into the clustering and is compared with the concepts representing both clusters, since reptile contains at least one value which is predictive of each. Appropriate value weights are decremented, since some of reptile's values differ from predictable values of each cluster, while the remaining values of each cluster are incremented. After decrementing values of the concept representing amphibian-2, the weight of the 'Fertilization' value is -1, necessitating that it be dropped as a predictable value of this cluster *and* that it be removed as a predictive value of the cluster. A concept representing reptile is then added to the clustering. This addition causes 3 clusters to be indexed by the value (unregulated), and it is removed as a predictive value of each cluster. The resultant hierarchy follows.



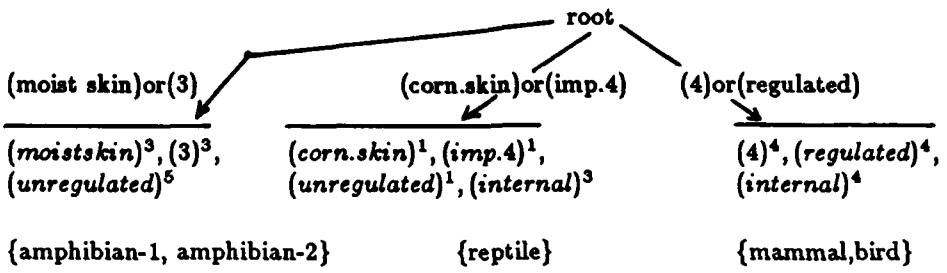
Next, the object bird is added to the clustering. This object has the value (internal) which is predictive of the clusters corresponding to amphibian-1 and reptile. Appropriate values of these two clusters are incremented and decremented and two predictable values, (moist skin) and (3), of the amphibian-1 cluster are dropped as both predictable and predictive values, as a result. A cluster corresponding to bird is then added to the clustering. The result of this is that (internal) indexes 3 clusters and is thus deleted as a predictive value of any cluster. This, in turn, leaves the cluster corresponding to amphibian-1 with no predictive values and it is removed from the clustering.



Next we incorporate 'mammal' which possesses predictive values of the cluster containing 'bird'. The weight of the (feathers) value is decremented, while its remaining values are incremented and mammal is added as a cluster.



If we now iterate through the input a second time, say in the order (amphibian-1,mammal,reptile,bird,amphibian-2), we obtain a clustering of the form[15]



As the example indicates, the clustering evolves slowly over a stream of input. Values which were previously deleted as predictive might be added back at a later time. UNIMEM assumes that the clustering will converge on a 'natural' structure after a large set of input. This property is not provable, and in general, UNIMEM's behavior appears difficult to characterize in any formal way.

5. Concluding Remarks

In the preceeding pages we have discussed methods of conceptual clustering as extensions to those of numerical taxonomy. The two approaches differ primarily in the way each class of methods represents similarity between objects and/or object groups. Methods of numerical taxonomy express similarity as a numeric value and these methods appear to implicitly assume that objects can be represented naturally in terms of continuously valued variables. We do not believe that the use of numeric representations of similarity can generally be used to naturally represent similarity between objects which are defined primarily in terms of categorical (i.e., discrete-valued) variables. In contrast to techniques of numerical taxonomy, the conceptual clustering methods we have discussed express the similarity between objects as a set of values common to all objects of an object group. In keeping with a long standing tradition in AI, methods of conceptual clustering assume that objects are represented only in terms of categorical variables. We cannot presume to know whether the particular methods of conceptual clustering presented in this paper would have great utility to researchers in data analysis - these methods, after all, represent initial work in the area of conceptual clustering. However, we feel that viewed as a process abstraction, conceptual clustering could envelop future techniques of considerable utility. In particular, concepts need not be restricted to

[15] Actually, after this second iteration through the input, a fourth node corresponding to 'mammal' would also be present. We would expect this fourth node to be eliminated after further iterations through the input.

sets of variable values common to all objects of some group. A number of alternative concept forms have been suggested.

In existing conceptual clustering techniques, objects are represented as variable value pairs, so that relationships between object values cannot be naturally represented. Work is presently underway [MIC83B, LEB83B] to devise conceptual clustering methods which permit clustering of *structured* objects. A structured object representation is one which explicitly represents the relationships which exist among values of an object. Languages suitable for structured object and concept representation include first-order predicate calculus and graphical representations such as semantic networks[QUIL68].

A second direction for extending present concept representations has been suggested by Michalski and Stepp [MIC83C]. They suggest including implication and equivalence as possible logical connectives used in concept representation. Lebowitz [LEB83A] discusses the relationship between 'predictiveness', as utilized in UNIMEM, and logical implication. UNIMEM attempts to construct clusters whose constituent members share a subset of (predictive) values which imply the presence of the shared remaining (predictable) values. UNIMEM's method of clustering is used in a program, IPP, which accumulates knowledge of terrorist events from newswire reports. The clustering obtained over an initial stream of reports is used to infer properties of future reported events, given only partial (predictive) information. By allowing implication and equivalence, a conceptual clustering technique can derive concepts which explicitly represent hypotheses or beliefs which can be tested on future observations.

Finally, concepts employed by present conceptual clustering techniques are roughly equivalent to sets of *necessary and sufficient conditions*,[16] which must be satisfied by all concept members. Existing conceptual clustering methods would probably have difficulty in identifying exceptional objects (i.e., *outliers*) or in dealing effectively with *noisy* data. This is probably due to the restrictions imposed by representing concepts as sets of necessary and sufficient conditions. Medin and Smith [MEDI81] discuss a *probabilistic approach* to concept representation in which probabilities or weights are associated with certain *modal* values of a concept. These measurements reflect the degree to which concept membership is dependent on each variable value. This form of concept representation subsumes representations equivalent to necessary and sufficient value sets, and we believe the increased flexibility of such a representation would allow conceptual clustering to more effectively deal with noise and exceptional objects. We believe this is an area in which statistical and probability theory can be brought to bear in order to insure methods which are theoretically sound.

[16] The use of variable-value set pairs instead of variable value pairs makes concept representations used by present systems somewhat more general than sets of necessary and sufficient conditions.

REFERENCES

[BRUN56] Bruner, J., Goodnow, J., Austin, G. *A Study of Thinking*, Wiley and Sons, New York, 1956.

[EVER80] Everitt, B. *Cluster Analysis*, Heinemann Educational Books, Ltd., 1980.

[FISH84] Fisher, D. *A Hierarchical Conceptual Clustering Algorithm*, Technical Report, Department of Information and Computer Science, University of California, Irvine, 1984.

[HART79] Hartwig, F. *Exploratory Data Analysis*, Series: Quantitative Applications in the Social Sciences, Sage Publications, Beverly Hills, California, 1979.

[LAN84C] Langley, P. and Sage, S. Conceptual Clustering as Discrimination Learning. *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (1984), 95–98.

[LEBO82] Lebowitz, M. Correcting Erroneous Generalizations. *Cognition and Brain Theory 5*, 4 (1982), 367–381.

[LEB83A] Lebowitz, M. Generalization From Natural Language Text. *Cognitive Science 7*, 1 (1983), 1–40.

[LEB83B] Lebowitz, M. Concept Learning in a Rich Input Domain. *Proceedings of the Machine Learning Workshop* (1983), 177–182.

[MEDI81] Medin, D. and Smith, E. *Categories and Concepts*, Harvard University Press, Cambridge, Massachusetts, 1981.

[MICH80] Michalski, R. Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and Algorithm for Partitioning Data into Conjunctive Concepts. *International Journal of Policy Analysis and Information Systems 4*, 3 (1980), 219–243.

[MIC83A] Michalski, R., and Stepp, R. Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 5*, 4 (1983), 396–409.

[MIC83B] Michalski, R. and Stepp, R. How to Structure Structured Objects. *Proceedings of the International Machine Learning Workshop*, University of Illinois at Urbana-Champaign (1983), 156–160.

[MIC83C] Michalski, R. and Stepp, R. Learning from Observation: Conceptual Clustering. Appearing in *Machine Learning*, Michalski, R., Carbonell, J., and Mitchell, T. (eds.), Tioga Publishing Company, Palo Alto, California, 1983.

[MITC82] Mitchell, T. Generalization as Search. *Artificial Intelligence 18*, 2 (1982), 203–226.

[NILS80] Nilsson, N. *Principles of Artificial Intelligence*, Tioga Publishing Company,, Palo Alto, Ca., 1980.

[ORAM73] Oram, R. *Biology: Living Systems*, Merrill Publishing Co., Columbus, Ohio, 1973.

[QUIL68] Quillian, R. Semantic Memory. Appearing in *Semantic Information Processing*, Minsky, M. (ed.), MIT Press, Cambridge, Mass., 1968.

END

FILMED

11-85

DTIC